DETECTING AND TRACKING HUMAN MOTION IN VARIANCE-BASED RADIO TOMOGRAPHY IMAGING

by

Shobhit Gupta

A thesis submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Science

School of Computing

The University of Utah

May 2013

The University of Utah Graduate School

STATEMENT OF THESIS APPROVAL

The thesis of **Shobhit Gupta**

has been approved by the following supervisory committee members:

Suresh Venkatasubramanian	, Chair	**10/29/2012** Date Approved
Neal Patwari	, Member	**07/06/2012** Date Approved
Sneha Kumar Kasera	, Member	**07/06/2012** Date Approved

and by **Alan Davis** , Chair of the Department of **School of Computing**

and by Donna M. White, Interim Dean of The Graduate School.

ABSTRACT

Currently, few methods exist to accurately model a human motion inside a monitored area. Most of the approaches that exist depend on some kind of boolean data from sensors that tell the presence or absence of person a at a given instant of time near a particular sensor. Using that information, some systems can then track a person across the area at different timestamps. Furthermore, for most existing approaches, the accuracy drops rapidly as the number of persons in the image increases. The sensors used in such settings are usually expensive. Not much work has been done to build a similar system based on inexpensive radio sensors. As there is no way for our radio sensors to provide information as to whether a person is present at a location, we need to extract it from the data using computer vision and machine learning techniques. However, it is not easy in such a system to model the noise component accurately. Therefore, we provide a probabilistic model to decide whether a detected blob is noise or an actual person. In our work, we exploit the fact that images do not change by much between successive timeframes and use this to detect and track multiple persons in a monitored area with a reasonably high accuracy. We use location and count of persons in historical images, and their similarity with the current image to calculate the new locations and count.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

The past two years at the University of Utah have been an exciting journey for me. Apart from the knowledge that I gained from the coursework by doing assignments and projects, it has been a great learning experience through my interactions with my peers, labmates, advisors, and all the people working as part of the CPS project.

I would like to take this opportunity to thank and express my gratitude towards my advisor, Dr. Suresh Venkatasubramanian. He has been exceptional in providing guidance and support throughout the course of this thesis. He always encouraged me to look for new methods and algorithms when one method did not work. He was very patient when the results were not as expected and encouraged me to start over and think about the problem by using a different level of details in the data that we used. He would also encourage me to participate and attend talks and sessions which might not have a deep impact on the thesis but have surely enhanced my learning and understanding about different aspects of computer science.

I would also like to thank Dr. Neal Patwari and Dr. Sneha K Kasera for serving on my advisory committee.

I would also like to thank Avishek, Parasaran, John, and Amir for helping me with all the questions I had about anything related to my thesis and otherwise.

Lastly, I would like to thank my friends and family who provided unconditional support and encouragement without having the slightest clue about the work I was doing.

CHAPTER 1

INTRODUCTION

1.1 VRTI Introduction

Variance-based Radio Tomography Imaging (VRTI) is a transmission-based imaging method that measures received signals on many different paths through a medium to locate moving people in an monitored area surrounded by inexpensive radio sensors. It is a device-free localization technique in which people do not need to carry Radio-frequency Identification (RFID) tags. It uses the signal strength variance caused by moving objects within a wireless network. There have been some methods proposed and applied to detect, locate, and track people who do not carry radio devices in an indoor environment [2, 3, 4, 5]. When motion occurs near a wireless link, some of the multipath components may be affected. We quantify the intuition that motion in spatial areas where many multipaths exist causes more variance of the Received Signal Strength (RSS). By combining RSS variance information for many links in a wireless network, not only can motion be detected, it can also be localized.

Each deployed node is capable of recording and sending RSS measurements. Each pair of nodes constitutes a link, and each link records a power reading in the region and broadcasts it periodically. VRTI measures the variance of the link's RSS values.

Each link's RSS measurement are believed to follow an ellipsoidal model [6] and the impact that an object can have on a link decreases as the person moves away from the link. The link impact also decreases as the distance between the two nodes forming the link increases. The amount of RSS variance and the link location where it is observed relates to the physical location of motion, and an image representing motion is calculated using measurements from different links in the experimental setup. The system assumes that all RSS variations are due to movement of different objects or are caused by noise in the environment. The use of RF as opposed to much higher frequency EM waves (e.g., x-rays), introduces significant non-line-of-sight (NLOS) propagation in the transmission measurements. In multipath environments, fading plays a significant role in the received signal strength of a wireless link. Small changes in the phase of a few multipath components,

even due to motion outside of the network area or small changes in node position, can dramatically impact the measured RSS. Fading effects are therefore a significant part of the noise statistics.

Objects moving near a node will usually cause larger fluctuations in RSS on a link than the same objects moving at positions far away from either node; hence, we use a weighting matrix in our model that denotes the impact each link has at a particular pixel location. If all links in the network are considered simultaneously, the system of RSS equations can be described in matrix form as

$$s = Wx + n \tag{1.1}$$

where s is the vector of variance of RSS measurements for each link, n is a Mx1 noise vector, W is a NxM weighting matrix where each W_{ij} represents weight at pixel j due to link i as

$$W_{i,j} = \begin{cases} \frac{1}{\sqrt{d_i}} & \text{if } \mathsf{d}_{ij}(1) + \mathsf{d}_{ij}(2) < \mathsf{d}_i + \lambda \\ 0 & \text{otherwise} \end{cases} \tag{1.2}$$

where d_i is the distance between two nodes in link i, $\mathrm{d}_{ij}(1)$ and $\mathrm{d}_{ij}(2)$ are the distances of the nodes in link i from pixel j, and λ is a parameter which controls the width of the ellipse. x is a Nx1 motion image to be estimated where each x_j represents activity at pixel location j as

$$x_j = \begin{cases} 1 & \text{if motion occurs in voxel j} \\ 0 & \text{otherwise} \end{cases} \tag{1.3}$$

Figure 1.1 illustrates the setup.

1.2 Background

1.2.1 Blob

A blob is an area of connected pixels having similar pixel intensity or having the same color. All pixels belonging to a blob are considered to be in the foreground. All other pixels are in the background. Binary images have background pixel intensity equal to zero and all foreground pixels as nonzero.

We can use blob analysis techniques to detect blobs in an image and extract information about parts of the image. Blobs are usually not of any definite shape and can be made up of any 2D shape in the image.

Blob analysis is a powerful tool to search for an object in the image. We can extract information about the image by calculating information about the size, location, and number

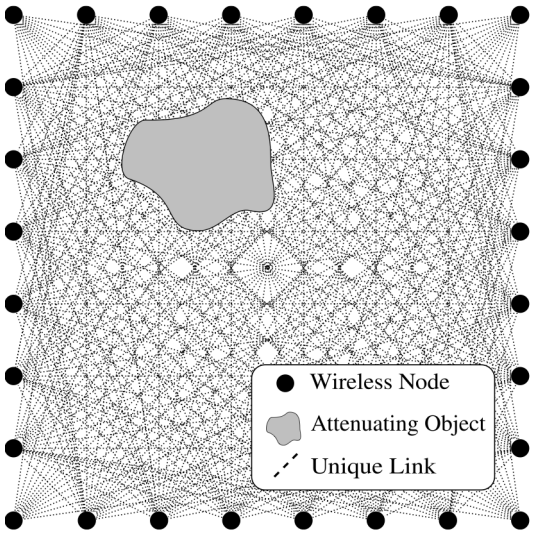

Figure 1.1. Image showing experimental setup

of blobs in the image. It also helps us filter out blobs that do not satisfy certain criteria. This helps us in eleminating blobs of no interest, thereby saving time in image processing. Blobs also help us in representing an image in compact blobs, which are easier to manipulate than the full image.

In our experiments, a person's footprint in an image can be viewed as a blob. So whenever we say blob in our setting, we will be referring to an area of connected pixels due to person's footprint or due to noise.

1.2.2 Thresholding

Thresholding enables us to select certain ranges of pixel values in grayscale and color images that separate the objects from the background. Thresholding converts an image into a binary image, with pixel intensity values of 0 or 1. This process works by setting all pixels whose value is below a certain value to 0, and setting all other pixel values in the image to 1.

1.3 Motivation

The motivation for the work comes from its high applicability in elder care systems, secure facilities, and emergency situations.

1.3.1 Elder Care

There have been efforts in developing technology that could be applied at elder care places to help elders live independently longer and also reduce burden on the medical

system. The focus is on monitoring their daily activities and interactions across the room with different individuals. Current efforts in this area use RFID tags, which are required to be worn all the time. Elders need to remember to wear the tags everyday. Other efforts utilize an audio and video feed from microphones, and cameras, which are against their privacy and extremely invasive. Hence, the requirement is to detect the presence of individuals without extracting a detailed video or audio feed.

VRTI helps us solve most of the above-stated problems. As the sensors are deployed surrounding the area, elders need not remember to wear any tags daily. By not capturing a detail of a person's activities, but rather a location that could give us an idea on their activities, we preserve their privacy and do not intrude upon their day-to-day life. We use historical patterns to decide whether an elder is spending too much time in an activity in which they should not and hence raise an alarm, thereby reducing the need of constant human watch.

1.3.2 Secure Facilities

Consider a secured facility guarded by human and automated systems. Any security breach at such facilities could pose a serious economic and social threat. Though highly popular, it is relatively easy to fool such a system. For example, when an authorized person swipes his or her batch to request access to enter the building, there might be enough time when the door is open to give an unauthorized personnel access and a chance to sneak in behind the authorized person. Similarly, a visitor may be required to be escorted at all times, but a visitor with malicious intent might escape while his or her host is distracted. Our proposed facility would be able to detect that two people gained access to the door when only one card was granted access, and that the visitor had strayed from the host. Theft of assets can be detected, even when the assets' active RFID tag has been disabled by the thief. Our systems will still track the person who was near the tag at the moment when it was disabled. It can also be used to prevent accidents. For example, if a person is standing in an unsafe zone, a machine could automatically be disabled if it's working in the same area as the person and could prevent accidents due to human presence. In the event of an emergency or evacuation, the system provides automatic reporting of occupants by room, which verifies and tracks occupants as they leave and enter the room. Those remaining are more quickly located by emergency response personnel. These applications might be enabled using standard active RFID; however, such systems do not provide high enough resolution, and only tagged people are detected and located. The proposed system

also detects and protects the people who are untagged. A human looking at an image can possibly tell the location of a person, but we aim to solve the problem where this process can be automated. An unauthorized personal or unauthorized activity could be detected automatically without any human intervention.

1.3.3 Emergency Situations

Conditions in an emergency situation are critical with many unknown parameters. Knowing the number of people trapped and their location in a fire or beneath debris or inside a void could prevent delay in rescue opetaions and could help save lives by providing them timely help. It would also help rescuers by providing them the location of victims in an emergency situation so that the right amount of resources can be targeted at the right place. This could also provide information about victims who cannot move or are unconcious and cannot call for help.

1.4 Thesis Statement

As part of this thesis, we develop a methodology to identify the number of people in an image, locate their respective positions, and then track their motion across the monitored area. We apply different techniques to solve this problem and show why some of them work and some do not work.

1.5 Thesis Contributions

1.5.1 Detect Number of Humans in the Image

The research shall enable detection of a number of humans on any grayscale image when used with the right initial threshold. We should be able to detect this number with high accuracy. Increasing the number of people to be detected in the monitored region does not impact the accuracy adversely as does most of the other systems studied.

1.5.2 Locate the Humans

Once we have identified the number of humans present in the monitored area, we provide a method to detect their location with high accuracy. We use computer vision techniques along with a history of past observations to find location the person's location with high precision. For detecting a person's footprint, we use opencv blob detection libraries which perform the connected component blob labelling procedure on the image.

1.5.3 Track the Humans

Once we have a method to detect a person's location at every time frame, we can use it to track him across the monitored area. We make sure to use past observation to correctly identify each person and assign the correct trajectory to each of them.

1.5.4 Statistical Approaches

We also show different statistical methods to approach the problem and why a particular class of methods does not work in the setting. We show a clustering-based approach and a scan statistic approach and show shortcomings for each approach for our current setting.

1.6 Challenges

1.6.1 Noise

Presence of nonliving objects and wireless networks might cause noise in the environment. Wave interference with nonliving objects and wireless networks inside the monitored area might cause the waves to bend and show activity where there is no activity in reality. Also, it may cause activity to disappear where there is actual activity. These unexpected variations in RSS are caused due to the multipath propagation of waves in an indoor environment. This could lead to wrong detection and hence could lead to incorrect action to be taken based on incorrect interpretation of results. Hence, looking at an image at a time and making all decisions using only a single image is likely to produce less accurate results compared to using a couple of images together and using last known correct interpretation to help with the current image. Noise can also be due to hardware but noise caused due to the above reasons is not easily quantifiable and is hence a challenge in solving the problem effectively.

1.6.2 Irregular Blob Shape

The blob shape and size are not consistent across experiments or even during the same experiment. This rules out any image-matching technique that could be used to extract human blobs from the image and then compare them against previously saved blobs for the same person or same kind of experiment. Also, the interference patterns of waves change at different locations in the monitored area. The blob intensity is higher when a large number of links are affected by it and hence the intensity is low at corners. Hence, extracting blobs from the image and comparing the shape and size with the previous image does not work accurately in a large number of cases. These cases might require an adaptive approach

where the blob extraction threshold could be adjusted based on the location of activity or based on the a feedback system using a previous confirmed image.

1.6.3 Multiperson Tracking

As the number of people in an image grows, it becomes more and more difficult to detect and track them effectively. This can happen as the number of people in a monitored area increases. Their corresponding blobs start overlapping, making it difficult to detect them based on that single image without using any historical data. Also, as new people enter and leave the area, we need ways to figure out whether it is noise or new activity in the area. Similarly, with more than one person in the area, we need to figure out when a person has exited the region and when he is not being detected due to noise. Furthermore, when multiple persons in an experimental setting start to interact by standing close together or crossing each other while walking, it is challenging to detect and report two persons because the blobs are combined and similar in shape and size to the one person blob. To detect and track multiple people in a region itself is a challenging problem, but the presence of noise adds to the complexity.

1.6.4 Integration with Current VRTI System

The latest OpenCV packages and libraries are only available for C++ and integrating and converting them in python for real time use is a challenging task. We use older OpenCV python libraries and try and make them with the latest code by doing modifications and upgrade wherever neccesary.

1.7 Outline of the Thesis

The rest of the thesis is organized as follows:

Chapter 2 describes related work.

Chapter 3 describes the experimental setup and types of data we deal with.

Chapter 4 discusses some statistical approaches that do not work well for our setting.

Chapter 5 discusses our current approach in detail and how and why we think that it works.

Chapter 6 discusses the results we have for detection and tracking experiments.

CHAPTER 2

SOME RELATED WORK

Several groups have been active in the domain of detecting activities by utilizing various sensing techniques deployed in a monitored area. Also, tracking is an established research field that may be addressed from various viewpoints. There have been works by Tsukiyama and Shirai [7] where each object is detected in the environment and then the objects that correspond to humans are discovered and tracked. Other works by Hogg [8] involve extracting edges from the images and then matching the edges against the edges of a human model. Machne learning techniques using classifiers to detect human presence have also been looked upon by Polana and Nelson [9].

Subtraction has also been used by subtracting the current image from the previous image in a pixel by pixel manner, using the intensity values [9]. A slightly different approach than this algorithm states using three consecutive images instead of two [10]. The result reflects movements between the images, unless the subject has the same intensity as the background. In a static scene, a base calibration background image without any humans may be recorded and used as a reference [11]. A more advanced version is to constantly update the background image while processing [12]. However, these methods assume that human motion is the only motion in the monitored area.

Other methods take advantage of thresholding techniques. By thresholding, it is easy to separate a person from the background [13, 14]. A related approach is to use infrared cameras. Thermal images can be obtained where the human is easily segmented through thresholding [15]. There have also been efforts by Ran Esheland Yael Moses [16] to use multiple cameras placed at an elevation and then used to detect heads to detect human presence.

Statistical approaches using characteristics of a group of pixels have also been used. Each pixel in the current image is compared to the statistics of the background image and classified as background or not [17]. More closely related to our work is a more advanced approach where the subject is modeled by a number of blobs with individual color and

spatial statistics. Each pixel in the image is then classified as associated with one blob based on its color and other properties [18].

Clustering-based approaches have also been common where each pixel is assigned to the closest cluster. The closeness in these situations is usually defined using a distance function which could be the difference in pixel location or the pixel intensity [19].

CHAPTER 3

EXPERIMENTAL SETUP

3.1 Setup Details

- The experimental region is a rectangular experimental region with or without immovable objects.
- Telosb radio nodes are set up inside the region in a rectangular fashion equidistant from each other. Nodes are also deployed on the region boundary. Some of the experiments also have nodes which are not deployed across a rectangular boundary. Two sample setups are shown in the figures below. Black dots in Figures 3.1 and 3.2 depict the telosb node locations.
- Every node talks to every other node, constituting a link. Each link reports the power readings and when a human crosses the link, the power reported drops. This helps in generating an image with the estimated location of human activity.
- One receiver node is connected to the computer, which receives data from each node and writes to a file on the machine.
- All transmitter nodes transmit data in a broadcast manner. Each node receives data from every other node and then the node transmits the full array of values, its own and the ones received from other nodes.

3.1.1 Image Data

- Each pixel in the image has a numerical value representing a certain color.
- Values range from 0 to several hundreds. Most values are between 0 and 8. Below are shown cumulative distribution functions (CDF) of pixel values for 0-person, 1-person, and 2-person images confirming the range of values a pixel can have. These can be seen in Figures 3.3, 3.4, and 3.5.
- Value of 0 denotes no human presence at that pixel location, represented by blue color in RGB or black in a grayscale image.

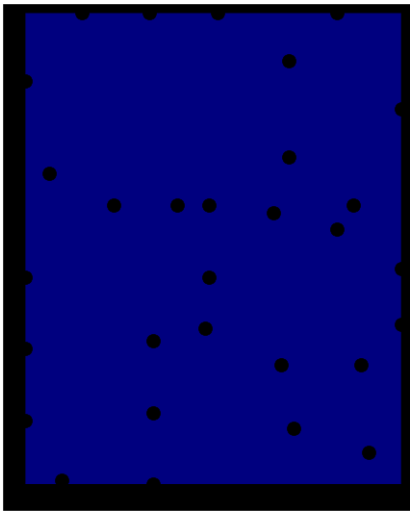

Figure 3.1. Nonrectangular node setup

Figure 3.2. Rectangular node setup

- The higher the value is, the higher is the confidence of a human presence. This is represented by red color in RGB or white color in a grayscale image.
- Black dots represent the location of sensor nodes.
- Figure 3.6 shows what the image for one person looks like. The blob in white shows the location of the person. Figure 3.7 shows the presence of two people in the region denoted by two blobs of lower intensity. For easier interpretation, the two blobs in the 2-person image have also been encircled with a red circle.

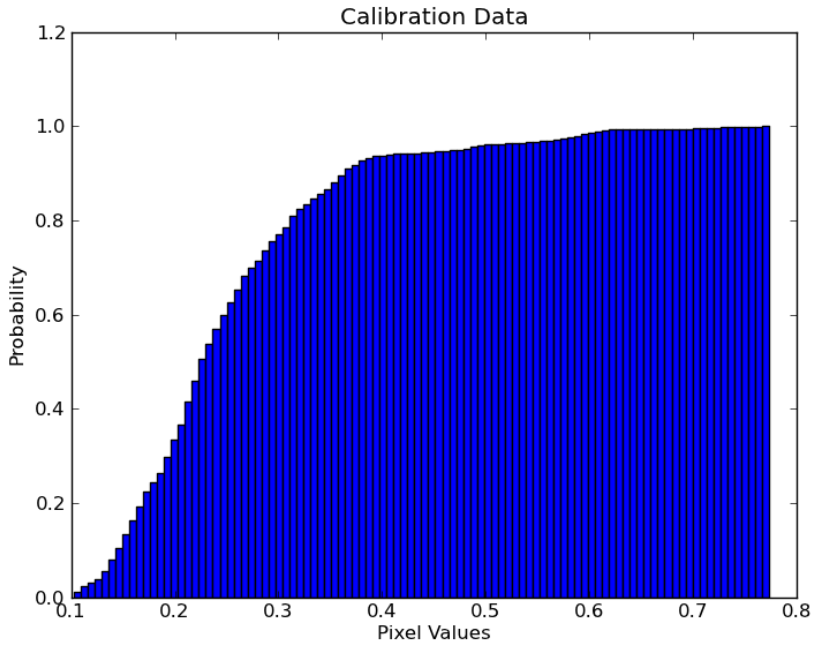

Figure 3.3. CDF of pixel values for 0-person images

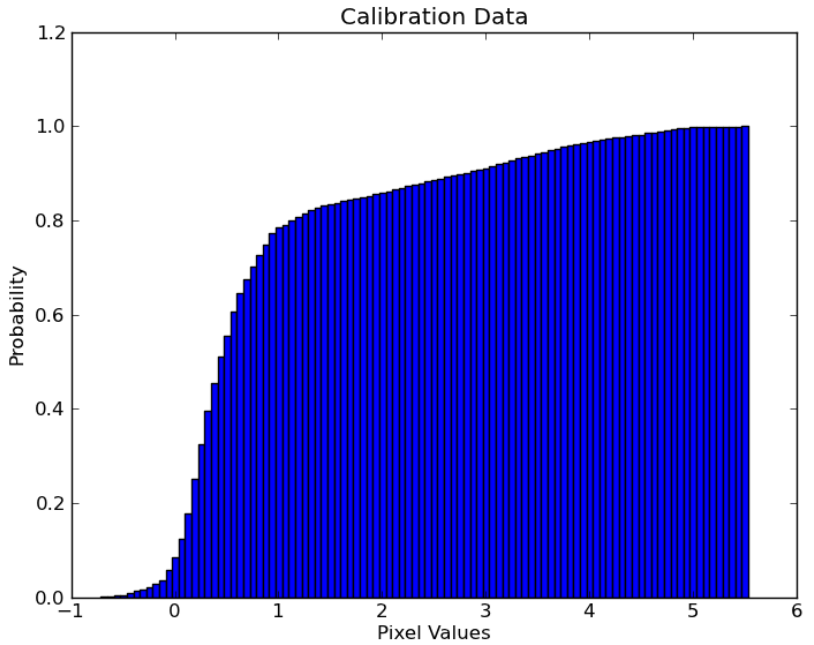

Figure 3.4. CDF of pixel values for 1-person images

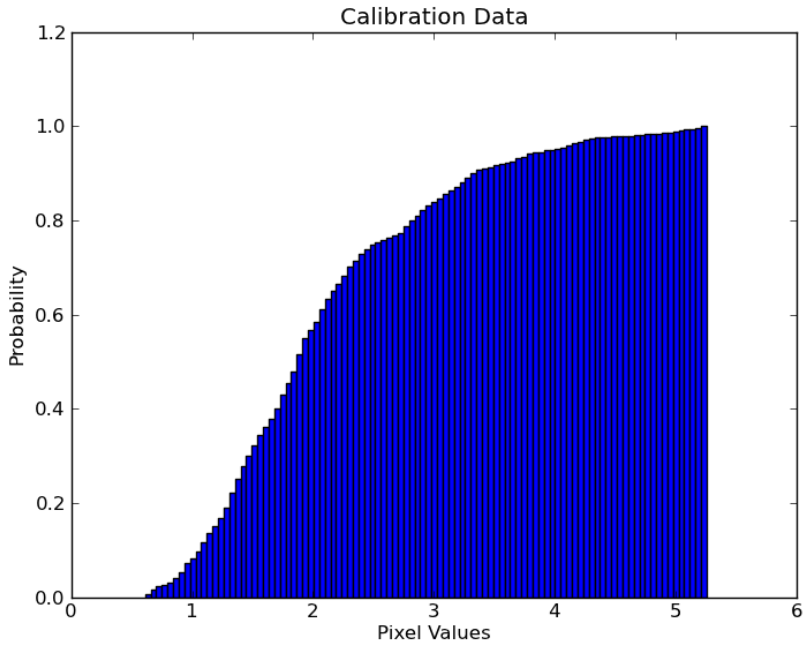

Figure 3.5. CDF of pixel values for 2-person images

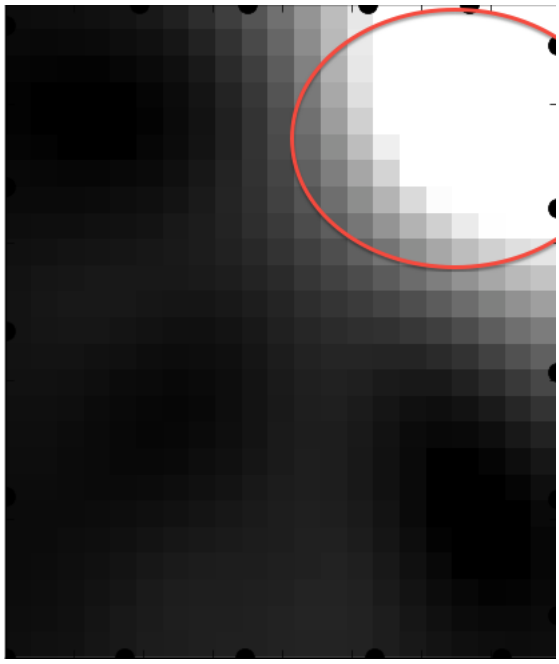

Figure 3.6. A grayscale image showing human presence with a white blob

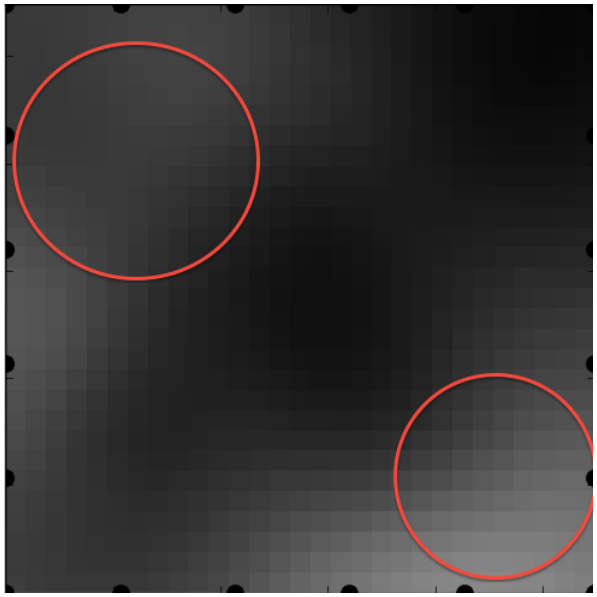

Figure 3.7. A grayscale image showing two humans denoted by two blobs

3.1.2 Link Data

- Link data are made up of raw RSS data received periodically from radio nodes in a cyclic manner.
- Data contain node ID from which the data were transmitted, followed by data that node received from all the other nodes, followed by an optional field for timestamp.
- These raw data are converted to decibels and used in image reconstruction along with the link weighted matrix.
- Figure 3.8 shows the probability distribution for raw RSS data.

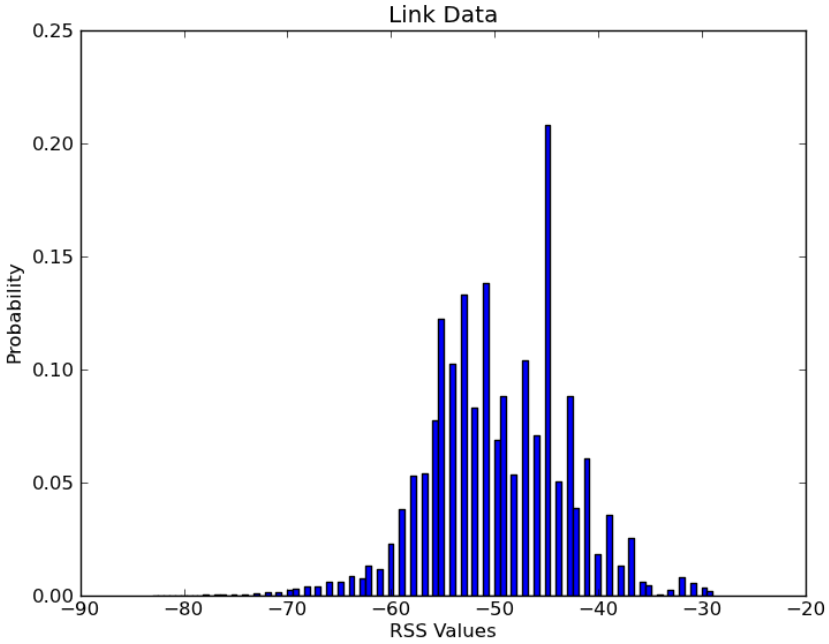

Figure 3.8. Probability distribution for raw RSS data

CHAPTER 4

ALGORITHMS THAT DO NOT WORK

4.1 Background

4.1.1 Support Vector Machines (SVM)

Support Vector Machines (SVM) are one of the most popular supervised classification techniques. Given a set of training points in space, each having a label, SVM tries to find a hyperplane with the largest margin that clearly separates the group of points according to their labels. This hyperplane is then used with a testing date for classification into one of the several labels. SVM can be used with different kinds of classifiers based on the data. If d-dimensional data are clearly separable using a d-1 dimension hyperplane, the classifier is called a linear classifier.

An example of linearly separable data can be seen in Figure 4.1 where rectangles repesent one label and the circles represent the other.

As we will see in some of our examples the data are not always linearly separable. For such cases, we use kernel functions. Kernel functions take original data and map them to a high-dimensional feature space where the data might be linearly separable. Figure 4.2 shows data in one dimension which are not linearly separable. Figure 4.3 shows same data are linearly separable when used with a quadratic kernel function.

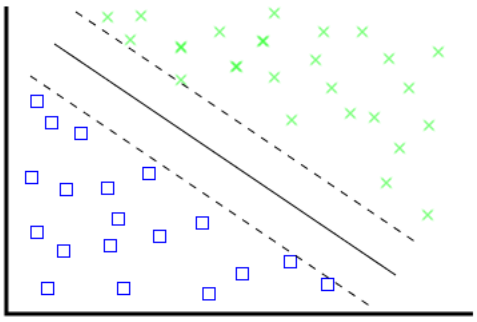

Figure 4.1. SVM linear classifier

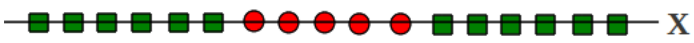

Figure 4.2. Nonlinearly separable data

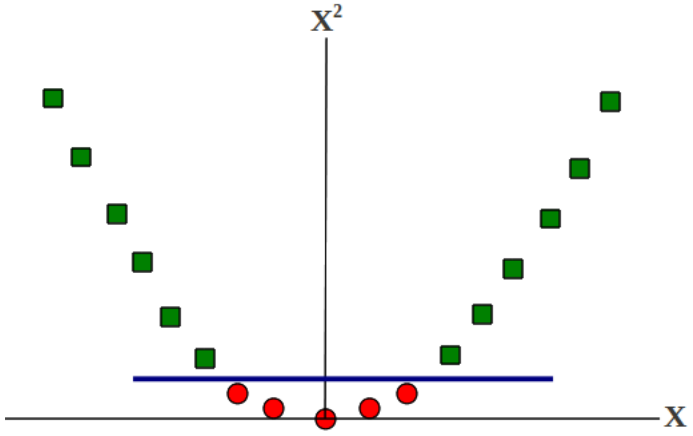

Figure 4.3. Data are linearly separable in higher dimension

In our work, we use a Radial Basis Function (RBF) Kernel which maps data to an infinite dimensional space. Using the kernel trick, the mapping need not be calculated explicitely. Rather, distance between two points can be calculated using the Kernel Function. In a RBF kernel, Kernel Distance between two points x and y is given by

$$K(x, y) = e^{-\gamma||x-y||^2} \tag{4.1}$$

4.1.2 K-Means

K-Means is a clustering algorithm which tries to partition data into K clusters where each point in the dataset is assigned to the closest mean. The input to the K-Means algorithm is the dataset, K, which is the number of clusters to be found, and initial seed of K centers, which are used as the initial mean for K clusters. The algorithm is as follows:

- Assign each point to the cluster with closest mean.
- Recompute mean for each cluster with new clutser assignments.
- Redo these steps until convergence.

K-means also requires a distance metric which is used to calculate distance from cluster means. In our work, we use Euclidian distance, which is given by :

$$d(x, y) = \sqrt{\sum_{d=1}^{D}(x_d - y_d)^2} \tag{4.2}$$

4.1.3 Spatial Scan Statistics

Spatial Scan Statistics is a statistical method to detect the maximum discrepancy area in spatial data. It detects local excess of events points and tests whether such an excess could have occured by chance and are randomly distributed or are part of some clustering. It is usually used in detecting disease outbreaks and epidemics.

Three main properties of the scan statistic are:

- Geometry of the area being scanned.
- The probability distribution generating the events or data points.
- Shape and size of the scanning window.

The window is thought of as an area of fixed size and shape, which then sweeps through the study area. As the window moves, it defines a collection of zones. These different zones overlap with each other and cover the whole area. Then, scan statistic is defined as the maximum likelihood ratio across all zones given by :

$$S_z = \left(\frac{max_Z L(Z)}{L_0} \right) \tag{4.3}$$

where L(Z) is the likelihood function for zone Z and L_0 is the likelihood function under null hypothesis. For a poisson distribution, the Log Likelihood Ratio (LLR) is given by:

$$LLR(z) = \left(\frac{c_z}{n_z}^{c_z} \right) \left(\frac{C - c_z}{C - n_z}^{C - c_z} \right) \tag{4.4}$$

where c_z is the number of events in zone z and n_z the expected number of cases in zone z, C is the total number of events, and N is the total number of expected events. Now, scan statistic is given by

$$S_z = max_z LLR(z) \tag{4.5}$$

We use 2-dimensional scan statistics by reading the image using a rectangular window and reporting rectangle where we find maximum discrepancy compared to base value. For each window, the method compares the likelihood of distribution inside the window with the distribution outside the window.

4.2 Fitting a Guassian Blob

We try to fit a gaussian blob to the image using the following two parameters:

- Scaled Height: We scale the total height (sum of values of all pixels) of the image by a scaling factor.

- Standard Deviation of pixel values: We calculate variance and standard deviation for each image.

Our belief that data is gaussian is strengthened looking at the following figure. Figure 4.4 is a 3D plot for a 2-person image where we show how values of various pixels change with respect to the mean. The process of selecting these pixels and the mean has been described below.

The plot clearly shows two peaks in data which represent the two persons identified by the method.

The reason we choose scaled height is because we want to give higher weightage to pixels which have high values and are close together. This is based on our observation that pixels located where a person is present have high values and the pixel values decrease gradually as we move away from the mean. Also, we do not want to give high weightage to pixels which have high values but are far away from the mean. Both these concerns have been adressed in our scaled height calculation algorithm described below.

- We pick the location of the highest valued pixel as mean(μ).
- We calculate the weighted variance of the image using this mean's location value and then multiply it with the value of the pixel. So we calculate variance as

$$\sigma^2 = |x - \mu|^2 * val(x) \tag{4.6}$$

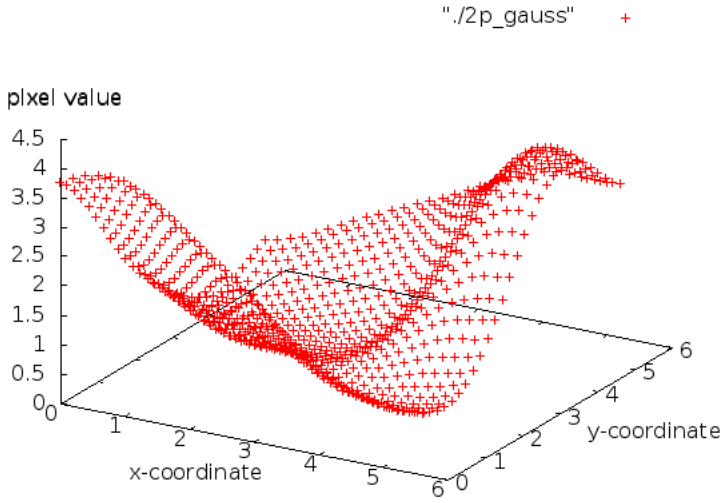

Figure 4.4. 3D plot showing pixels with 2*standard deviation distance of mean

where x is the location of the pixel and val(x) is the value of that pixel.

- For all pixels situated less than 2 Standard Deviation from the mean, we calculate height as $\frac{e^{-\frac{|x-\mu|}{\sigma^2}}}{2\pi\sigma^2}$ where x is the location of the pixel and σ is the standard deviation.
- The sum of heights for all such pixels gives us the scaling factor(sf).
- Now, scaled height is given by $\frac{\sum x_i}{sf}$.
- For every image in our experiment, we calculate this scaled height and the variance parameters.

Now, data are calculated from several 0-person, 1-person, 2-person images and plotted on a graph. X-Axis of the graph is the standard deviation and Y-axis is the scaling height. Every point in the plot is an image. We do this for every image and in the end, we see a separation between 0-person and 1-person figure, as shown in Figure 4.5, and in 0-person, 1-person, and 2-person images, as shown in Figure 4.6.

4.2.1 Support Vector Machines (SVM)

We use SVM libraries provided by libsvm software package [20]. We use a RBF kernel with SVM and train it using labeled data where data are 2-tuple value <scaling height, standard deviation> for each image. Then given test data in the same 2-tuple format for unlabelled images, SVM would perform the classification task of predicting the number of persons in the image. Tables 4.1 to 4.3 show results after we ran this method on different experiments. Table 4.1 shows results when 10% of the data was used as the training data. Table 4.2 shows results when 20% of the data was used as the training data and Table 4.3 shows results for 0-5 person data with a different amount of training data.

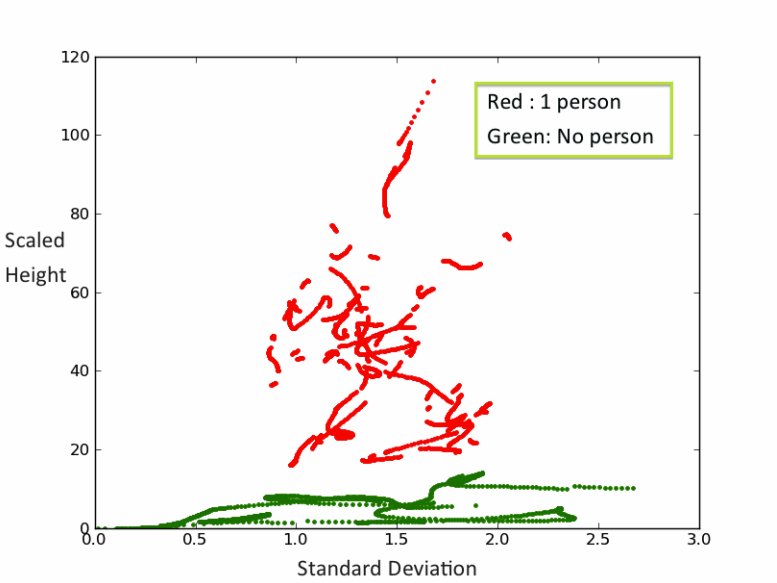

Figure 4.5. Image showing 0-person and 1-person plot

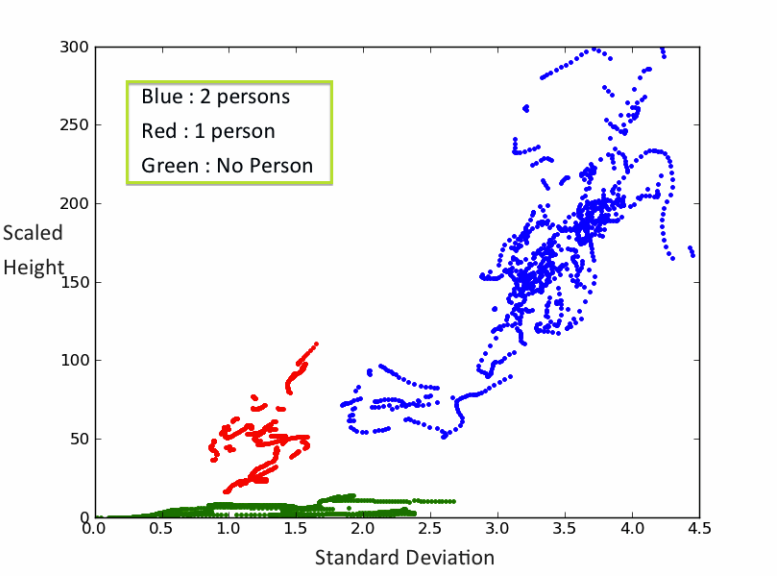

Figure 4.6. Image showing 0-person, 1-person, and 2-person plot

Table 4.1. SVM Prediction Results with 10% Training Data

Name Of Experiment	No of Persons	% Prediction Accuracy
0-1 Test 1	0,1	100
0-1 Test 2	0,1	100
0-1 Test 3	0,1	99.40
0-1-2 Test 1	0,1,2	99.8
0-1-2 Test 2	0,1,2	93.67
0-1-2 Test 3	0,1,2	93.89
0-1-2-3 Test	0,1,2,3	87.54
0-1-2-3-4 Test	0,1,2,3,4	85.26

Table 4.2. SVM Prediction Results with 20% Training Data

Name Of Experiment	No of Persons	% Prediction Accuracy
0-1 Test 1	0,1	100
0-1 Test 2	0,1	100
0-1 Test 3	0,1	99.52
0-1-2 Test 1	0,1,2	99.86
0-1-2 Test 2	0,1,2	94.34
0-1-2 Test 3	0,1,2	94.5
0-1-2-3 Test	0,1,2,3	87.99
0-1-2-3-4 Test	0,1,2,3,4	80.08

Table 4.3. SVM Prediction Results for 0,1,2,3,4,5-person Experiment

% Training Data	% Prediction Accuracy
10	78.23
20	69.73
33	79.36
50	77.9
80	80.55

Then, we use this prediction to feed it to the K-means algorithm and find the possible location of persons. The whole process is described completely using a flowchart shown in Figure 4.7.

K-means needs a set of points and a distance metric to correctly classify points into the clusters. We can either cluster the pixels by their value or their location. None of the two methods serve our purpose. So we calculate the number of pixels that should identify a person(k) and multiply by the number of persons we sense in the image(n). Then, we find out k*n pixels with the highest values in the image and give their location to K-means for clustering.

K-means output for the 1-person image is shown in Figure 4.8 and for 2-person in Figure 4.9. In the images, the white cross(x) denotes the estimated location of the person.

4.2.2 Problems

While SVM does a remarkable job in predicting the number of persons in the image, our task also requires us to locate them precisely. Hence, the accuracy of our method depends on SVM + K-means together. While the SVM + K-means method works well for images where the blobs are clearly separated and are of equal size. It falters when the blob size differs a lot. It might calculate the wrong results if the top k*n pixels correspond to less than k persons. As shown in Figure 4.10, the top k*2 pixels correspond to a single blob and we are not able to detect the second blob.

4.2.3 Next Steps

We need a method in which we do not need to choose the pixels to cluster ourselves, but the method should do it for us. As the blobs representing the persons are not of fixed size or shape, we cannot easily identify the number of pixels required to represent a person. A low value for this number would result in more pixels to be selected from the same blob and result in less than the required clusters. A high value for this number would result in pixels being selected from different blobs even for a single person, and thereby merging the clusters and resulting in less than the required clusters.

4.3 Spatial Scan Statistics

- Finds geographic clusters in 2D which exhibit maximum discrepancy when compared to a baseline value.

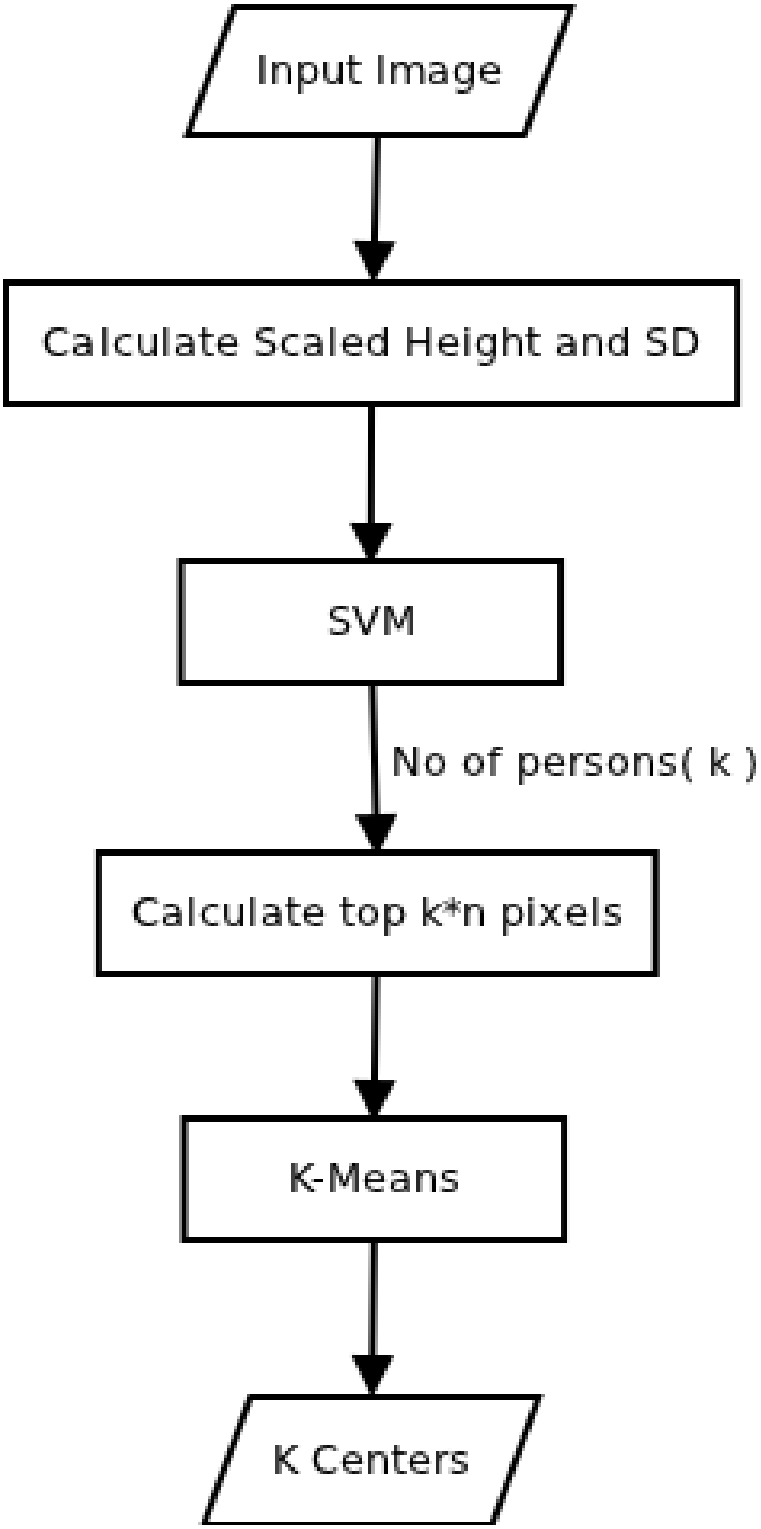

Figure 4.7. SVM-Kmeans flowchart

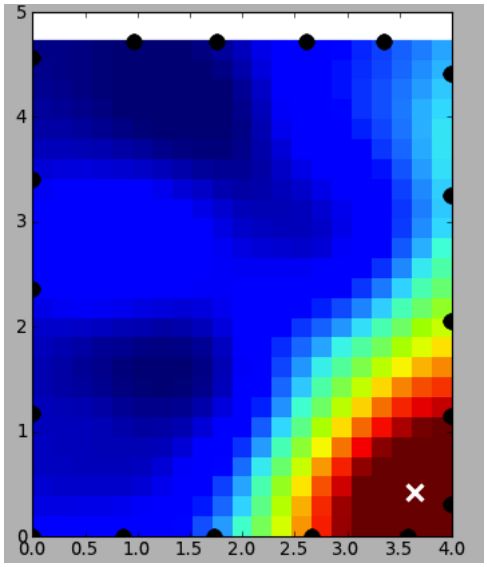

Figure 4.8. K-means: Image showing 1-person output

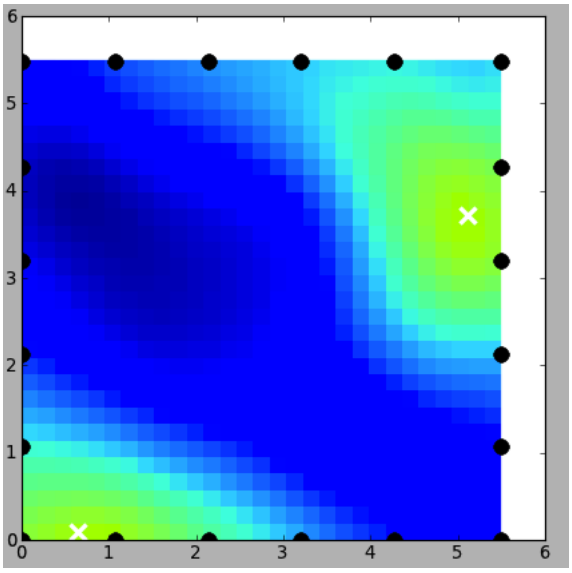

Figure 4.9. K-means: Image showing 2-person output

- Define a rectangular window and a baseline value and for each rectangle, calculate the Log Likelihood Ratio of the values being generated by a distribution.
- Scan statistic is the maximum LLR over all rectangles.
- As scan statistics is designed to find a single maximum discrepancy area, after finding the first such area, we replace its value with the baseline value and repeat the process to discover more such regions.

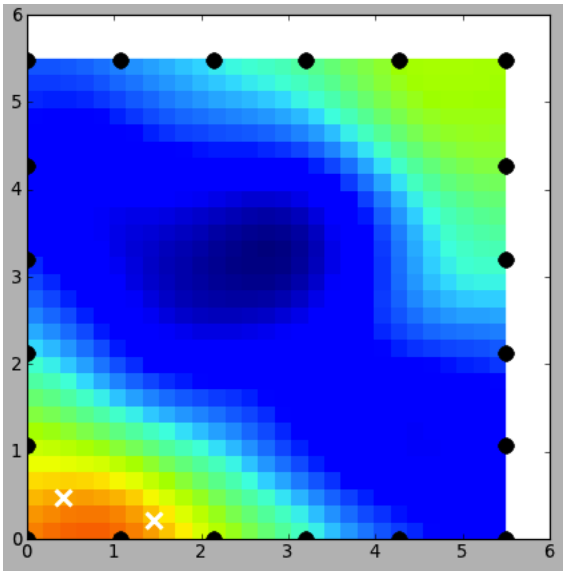

Figure 4.10. K-means: Image showing wrong 2-person output

Figure 4.11 illustrates what a 3-person image looks like when run through spatial scan statistics code. Only the rectangle's top right and bottom left corners are shown. The first located blob is displayed using white crosses, the second one is displayed using red crosses, and the last one has green crosses.

4.3.1 Problems

Spatial scan statistics works for unequal sized blobs and hence solves one of the problems we faced with the SVM + K-means technique. However, it still cannot handle blobs which are close together. In such a case, it might merge them into a single large maximum discrepancy area. Or it might show two maximum discrepancy areas where there should have been one. As shown in Figure 4.12, spatial scan statistics only detects one of the three blobs correctly and displays two blobs where there should have been only one.

4.3.2 Next Steps

We have observed that looking at only the current image is not sufficient and we need a method that can use information from previous images to make decisions for the current image. So we try to use computer vision techniques now instead of statistical techniques, which might be able to provide us with more information about the image and also provide us with tools to manipulate parts of the image as we desire.

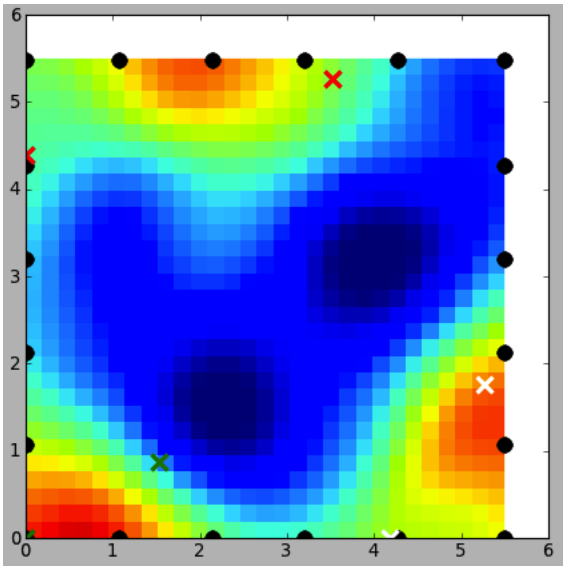

Figure 4.11. Spatial scan statistics: Image showing 3-person output

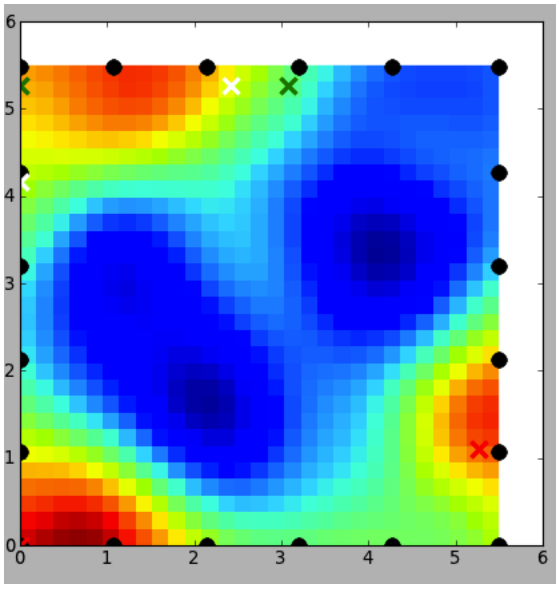

Figure 4.12. Spatial scan statistics: Image showing wrong 3-person output

CHAPTER 5

AN ALGORITHM THAT WORKS

5.1 Background

5.1.1 OpenCV

OpenCV is an open source BSD licensed library designed to help with problems related to Computer Vision. It includes several hundreds of computer vision algorithms. It also includes some static machine learning libraries. It has been optimized and intended for real-time applications. Some of the features provided by OpenCV are:

- Image manipulation
- Image and video input/output
- Object recognition
- Blob detection
- Matrix and vector manipulation

Some applications of the OpenCV library are Human-Computer Interaction (HCI); Object Identification, Segmentation and Recognition; Face Recognition; Gesture Recognition; Motion Tracking, Ego Motion, Motion Understanding; Structure From Motion (SFM); Stereo and Multi-Camera Calibration and Depth Computation; Mobile Robotics.

5.1.2 Threshold

We use the OpenCV blob detection library on this image, performing connected component labeling on the binary image. It also provides functions to manipulate, filter, and extract results from the extracted blobs.

The blob detection library provides two key functionalities:

- Extract 8-connected components in binary or grayscale images. These connected components are referred to as blobs.

- Filter the obtained blobs to get the interest objects in the image. This is performed using the Filter method from CBlobResult.

The library is thread-safe if different objects per thread are used.

5.1.3 Blob Detection Algorithm [1]

A binary image is scanned from top to bottom, left to right each line and the connected components are labelled using the following four operations.

- On encountering an external contour point A, make a complete trace of the contour until we return to A, as shown in Figure 5.1(a). All points on this contour are assigned a label the same as A.
- On encountering a labelled external contour point A', follow the scan line to find all white pixels and label them as A', as shown in Figure 5.1(b).
- On encountering an internal contour point B, B is given the same label as the component it is part of. Then, the internal contour containing B is traced and all points on this contour are labelled with the same label as B, as shown in Figure 5.1(c).
- On encountering a labelled internal contour point B', follow the scan line to find all white pixels, and assign them the same label as B', as shown in Figure 5.1(d).

These steps can be represented by three logical steps where each step deals with:

- newly encountered external point and all points on that contour
- newly encountered internal point and all points on that contour
- all white points not dealt with in the first two steps.

5.1.4 Jaccard Similarity Index

Jaccard Similarity Index is a popular statistic used to find the similarity and diversity metric between two sets. The Jaccard Index is 0 if the sets are disjoint and is 1 if the sets are identical. The value is close to 1 for similar sets. The Jaccard Index is calculated by dividing the intersection of the sets with the union of sets. The Jaccard Similarity Index between two sets A and B is given by:

$$J(A,B) = \frac{A \bigcap B}{A \bigcup B} \tag{5.1}$$

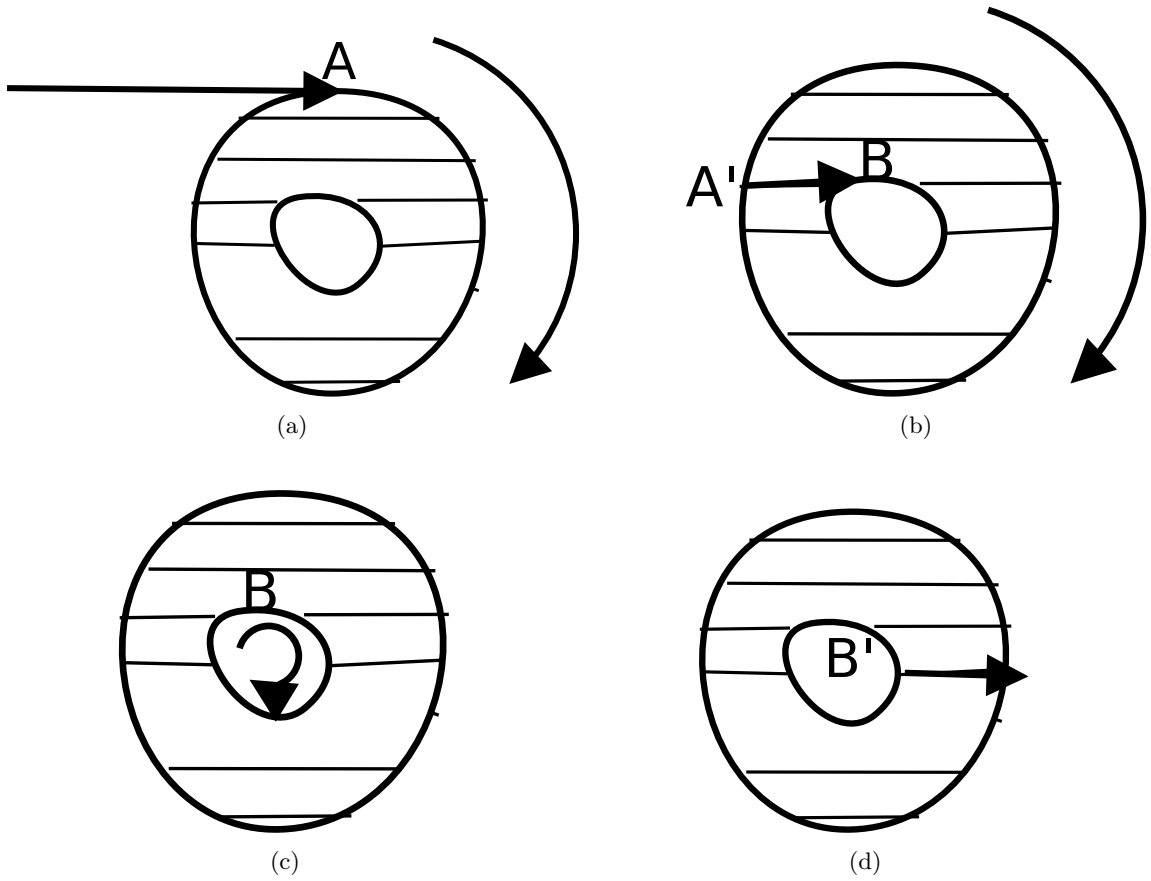

Figure 5.1. Blob detection algorithm

5.2 Global Threshold

We use a blob detection library with a random subset of 10% images from different 1-person and 2-person experiments and find the largest interval for these images that gives us the correct number of blobs. The correct number of blobs for these images is already known and fed into the code along with the image. Then, we calculate the intersection of intervals across all these images.

We then use this threshold interval with all images and calculate the number of blobs in each of the images from different experiments. We compare them against the expected number of blobs. We report success if the number of blobs calculated is the same as the expected number of blobs for at least one of the values of the threshold and return failure otherwise. Figure 5.2 represents the original grayscale image. Figure 5.3 shows the output image when the code is run with a 2-person image. The blob in red is the first person's footprint in the image and the green blob represents the second person.

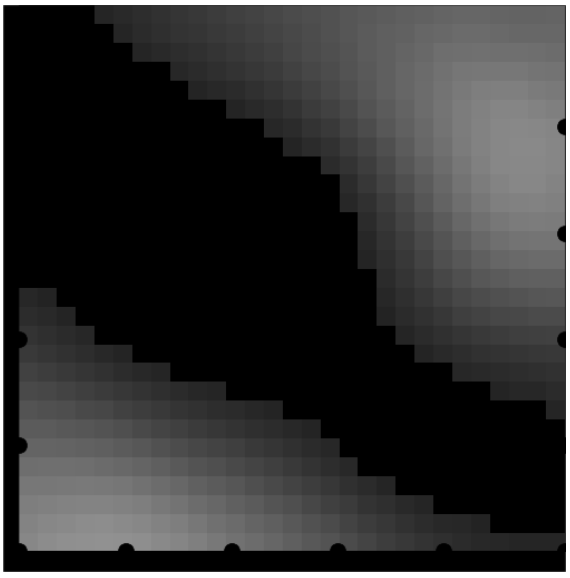

Figure 5.2. A grayscale image for a 2-person experiment

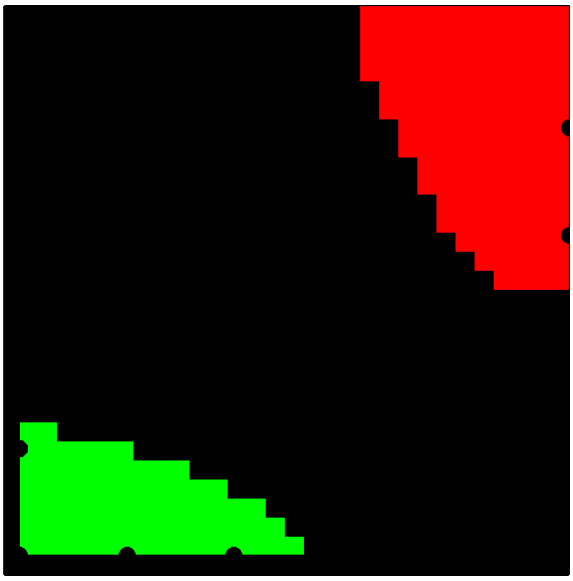

Figure 5.3. Image output from opencv using fixed threshold

5.2.1 Problems

Not all images have the same threshold. Wave interference patterns might cause a small noise to show up in some parts of the image and the threshold might need to be adjusted for the same. Also, similar wave interference patterns might cause actual activity to disappear and threshold needs to be decreased for this. Figure 5.4 and Figure 5.5 show the case when a global threshold does not work with a 2-person image.

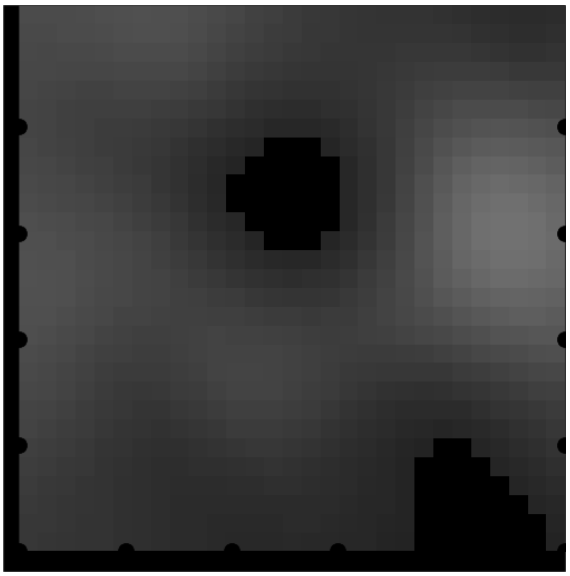

Figure 5.4. A grayscale image for a 2-person experiment

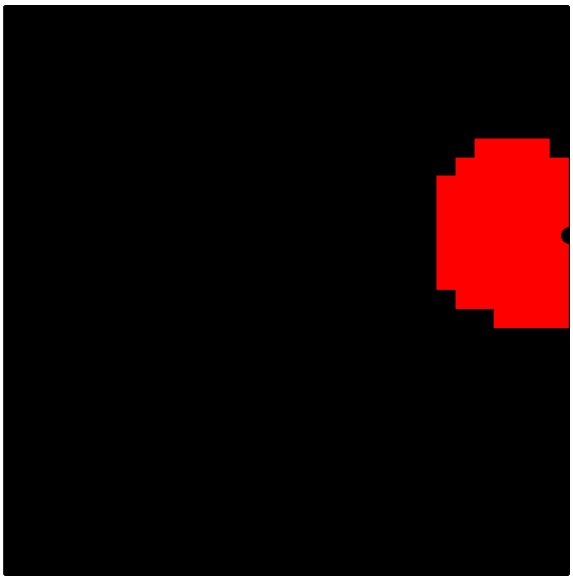

Figure 5.5. Image output from opencv using fixed threshold

5.3 Learn Threshold on the Fly

In this method, we supply code with an image and the expected number of blobs. We check the number of blobs in the image against the expected number of blobs. We increase the threshold if the number of blobs in the current image is more than the expected number of blobs and decrease when it is less than expected. If we are able to find a threshold for which we can detect the right number of blobs, we declare success else, if we do not find

any threshold at which we have the right number of blobs, we declare failure. We save the threshold calculated for the current image and use it as the starting threshold for the next image.

5.3.1 Problems

1. Learning threshold dynamically for every image yields better results in terms of accuracy, but then we might have to look at all possible threshold values in order to reach the right value.

2. There might be cases when it is not possible to get even a single threshold value which will result in the correct number of blobs.

3. There is no way to know the expected number of blobs. In an earlier method, we were using a global threshold and reporting the number of blobs as we detect them. However, in this method we do not know what is the expected number of blobs and hence, it is difficult to know where to stop. We only specified the expected number of blobs in our tests to test the accuracy of the method when the expected number of blobs is given.

5.4 Learn Threshold Using History

We believe that the number of persons in an image should not change too much from the previous time frame. Hence, we use a history of images and assume that the number of blobs in the current image is the majority of what we have in historical images. This method gives us the expected number of blobs to look for in the current image. And if indeed the number of blobs has changed, gradually our majority will change to reflect this.

We use a starting threshold and start building history; then, when every image comes in, we look at our history and try to find out the number of blobs to expect. Once we have the number of blobs, we vary the threshold to correctly calculate the blobs and report the results. This method solves our problem of finding out the expected number of blobs.

5.4.1 Problems

- It suffers from the problem of cases when it is not possible to get even a single threshold, which will result in the correct number of blobs.

- If the expected number of blobs in the current image is less than what we have in the previous image, we have no way to know whether a person has moved out or we need to lower the threshold.

- If the expected number of blobs in the current image is more than what we have in the previous image, we have no way to know whether a person has entered the monitored area or it is noise.

5.5 Use Location and Count as History

Until now, we have been using thresholding for extracting blobs from an image. We also tried using the majority from history to figure out the expected number of blobs. However, knowing the number of blobs alone does not suffice in some cases. It helps to know the location of blobs in the historical image as well.

This is due to the intuition that the images will not change too much between successive time frames and knowing the location of the blobs in the previous image can help us decide whether a new discovered blob is a person or noise.

So the method says that if the code shows a different number of blobs than expected, then maybe somebody moved out or somebody moved in or two persons are close together and their blobs have joined together. If we try decreasing the threshold and see that an extra blob appears where there was not any the last time, we make note of it but ignore if for the current image. Every time such a blob appears, we update its probability of being noise. When the probability of the blob being noise decreases below a certain threshold, we declare it as a person. We believe that a noise blob would not be as persistent as if there is a person in the region. Figure 5.6 shows the algorithm with the help of a flowchart. The algorithm steps are discussed in details below.

Including the location in the code helps us identify each of the above cases with more confidence. In order to use the location, we calculate a Jaccard Similarity index of each blob in the current image with each blob in the historical image. Any blob with more than 0.1 Jaccard similarity is declared to have been identified as corresponding to a previous blob. If two blobs in the current image have Jaccard similarity of more than 0.1 with the same blob in the historical image, we declare that the historical blob has been split due to low threshold and try to merge these two blobs. Whereas if one blob in the current image has Jaccard Similarity of more than 0.1 with two blobs in the historical image, we declare that probably the two blobs in the previous image have merged together and need to be split. Based on the our algorithm, we take one of the five decisions:

- **Update Threshold:** If there is some blob in the historical image which does not have Jaccard Similarity greater than a certain threshold with a blob in the current

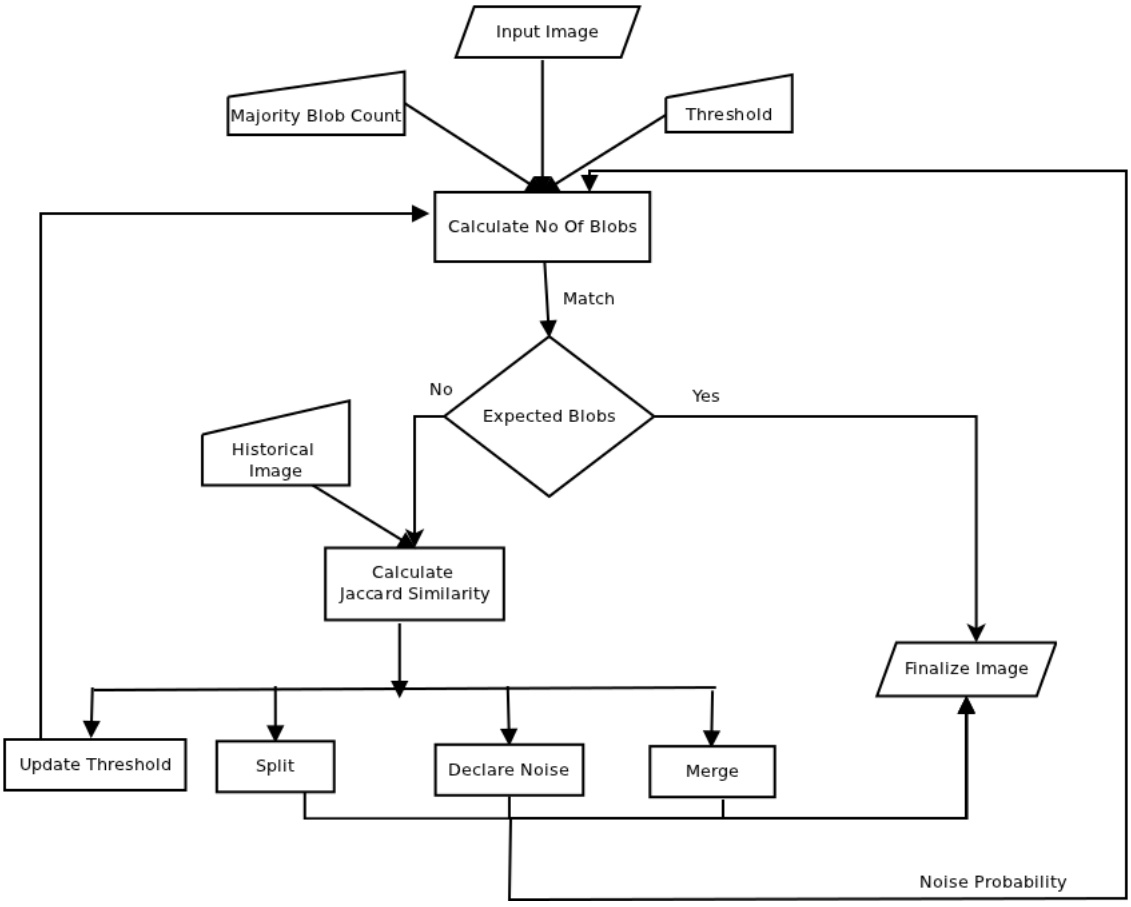

Figure 5.6. OpenCV algorithm flowchart

image, we believe that some blob might have dissapeared due to high threshold. So we decrease the threshold and perform all the steps again with the updated threshold.

- **Merge Blobs:** If there is some blob in the historical image which has a Jaccard Similarity greater than a certain threshold with two blobs in the current image, we believe that the old blob might have split into two. At this point, we try and merge the two blobs to make a single bigger blob. We also decrease our threshold by a fixed configurable number as we believe that our current threshold is too high and have resulted in blobs to split apart.

- **Split Blobs:** If there is more than one blob in the historical image which has a Jaccard Similarity greater than a certain threshold with a blob in the current image, we believe that the old blob might have merged into a single blob. At this point, we try and split the blob to make two smaller blobs. We also increase our threshold by a

fixed configurable number as we believe that our current threshold is too low and has resulted in blobs merging together.

- **Noise:** If there is no blob in the historical image which has a Jaccard Similarity greater than a certain threshold with a blob in the current image, we believe that this is a new blob that appeared in the current image. We take a conservative approach and declare the new blob as noise with a certain probability. If we encounter this blob again in the next time frame, we update its probability of being noise. When this probability falls below a value, we declare that this blob can no longer be noise and declare it as a valid blob. We also increase our threshold by a fixed configurable number as we believe that our current threshold is too low and has resulted in extra blobs to be detected.

- **Finalize Image:** If we take one of these three decisions, MERGING, SPLITTING, DECLARING NOISE, we finalize the image and replace the historical image with this new updated image.

5.5.1 Update Threshold

We believe that images should not change much during successive timeframes. However, sometimes due to wave interference patterns, it might happen that our system shows little activity in an area which has normal activity. In such cases, due to a high threshold it is possible to filter out this activity. Hence, we try to figure out such a situation using Jaccard Similarity and then decrease the threshold to force such small activity to show up. We keep on doing so until we are confident that we have not missed any activity due to a high threshold. Figure 5.7 and Figure 5.8 show how updating the threshold changes the number of people detected in an image. In the shown image, we expected two people in the image, but using the threshold from the previous image, only one was detected. So we lower the threshold until we find the expected number of blobs or we hit a lower limit of threshold and declare the image as not classified or failure.

5.5.2 Merging

We have implemented merging of two blobs in three orientations. In each of the three orientations, we try to find out four coordinates, two on each of the blobs, which form the four corners of a quadrilateral. Then, we set the values of the pixels on the boundaries of this quadrilateral to complete the merge. The three orientations are:

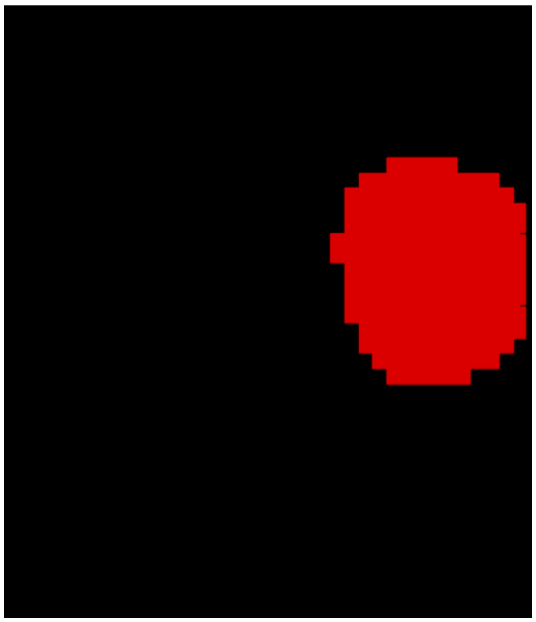

Figure 5.7. Image showing one blob before applying threshold

- **Vertical Merge:** When the lowest point of one blob is above the highest point of another blob. An example of vertical merge is shown in Figure 5.9 and Figure 5.10. On the top is a sample image with three blobs; at the bottom we see two of them have merged together and now only two blobs remain.
- **Horizontal Merge:** When the rightmost point of a blob is to the left of the leftmost point of another blob. An example of horizontal merge is shown in Figure 5.11 and Figure 5.12. On the top is a sample image with three blobs; at the bottom, we see two of them have merged together and now only two blobs remain.
- **Overlapping Merge:** When neither of the above conditions hold true.

5.5.3 Splitting

For splitting a blob into two smaller blobs, we use the historical blob information. We use the two blobs from history which should resemble the blob in the current image. We check the historical blobs' pixels and compare them against the blob in the current image. We remove all pixels from historical blobs which lie above, below, to the left of, or to the right of the blob in the current image.

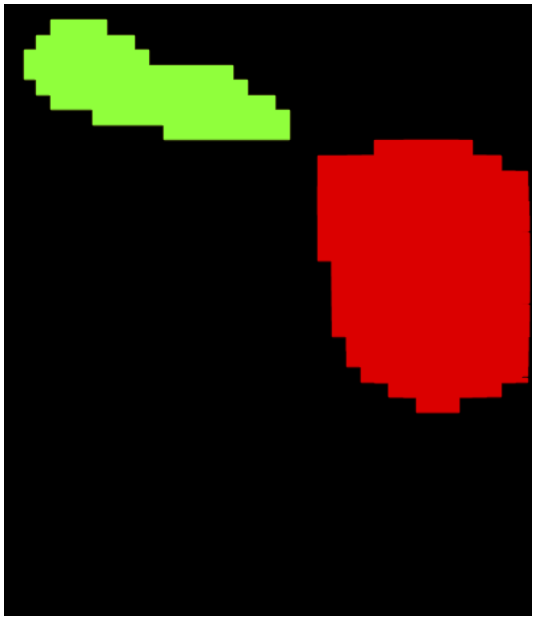

Figure 5.8. Image after updating threshold

Two consecutive images from the 2-person experiment are shown in Figure 5.13 and Figure 5.14. Next are shown the images as a result of applying the algorithm which detects that the second image should have two blobs and the blob should be split, as shown in Figure 5.15 and Figure 5.16.

5.5.4 Noise

The images from the 1-person experiment are shown in Figure 5.17 and 5.18. Figure 5.17 shows two blobs, but when Jaccard Similarity for both blobs was calculated with the previous image′s blobs, we detect that one of the blobs is extra and was not present in the previous image. We declare it noise with a probability p. Figure 5.18 shows the image finalized after declaring one of the blobs as noise.

5.5.5 Challenges

- **Splitting Blobs:** The process of splitting the blobs is not trivial. As the shape of the blobs is very irregular, it is a tough task to figure our the exact corordinates from where we want to split the blob. Also, calculating the exact line of split is a difficult task. Advanced computer vision and image processing techniques might yield better results for splitting blobs.

- **Merging Blobs:** The process of merging the blobs is not trivial. As the shape

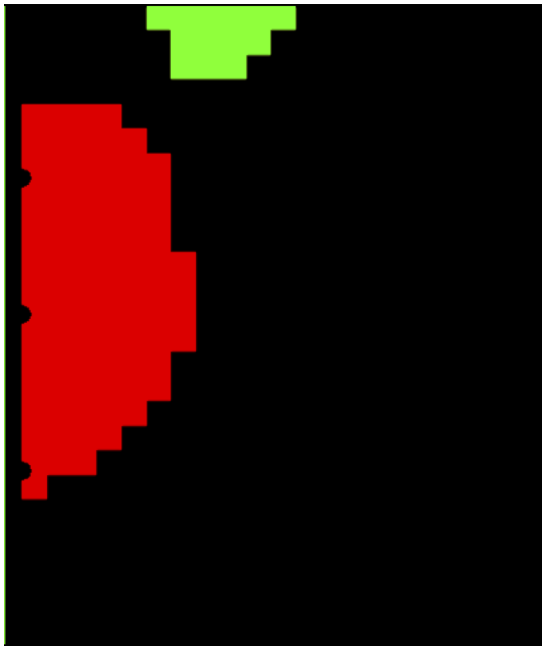

Figure 5.9. Image showing two blobs before vertical merge

of the blobs is very irregular, it is a tough task to figure our the exact coordinates from where we want to merge the blobs. Also calculating the exact line of merge is a difficult task. Advanced computer vision and image processing techniques might yield better results for merging blobs.

5.5.6 Calculating Location

We use the following three methods for location calculation of a person.

- **Bounding rectangle center:** In this method, we calculate a bounding rectangle for the blob we discovered. Then we calculate the center of the bounding rectangle and report that as the location of the person.
- **Centroid of blob:** We location all points lying inside the blob. Then we calculate the centroid of the blob and report that as the location of person.
- **Highest valued pixel:** In this method, we return the highest valued pixel in the blob as the person's location.

In the Figure 5.19, all three methods have been used to calculate center. The blue dot represents the bounding box method, the white dot represents the centroid method, and the black dot represents the maximum valued pixel method.

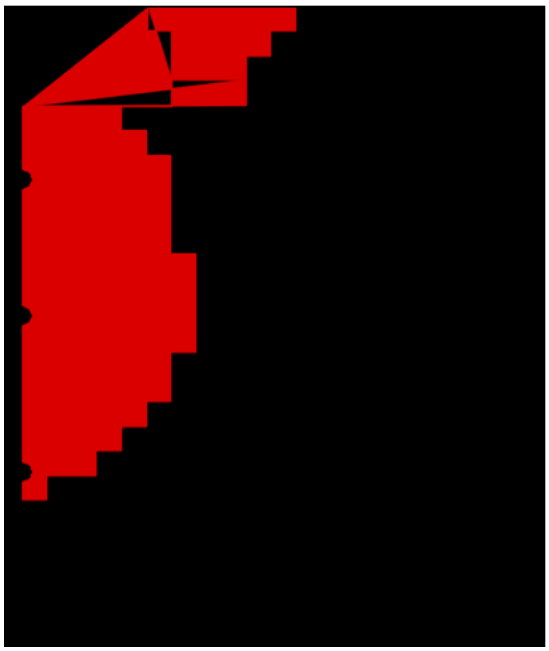

Figure 5.10. Vertically merged blobs

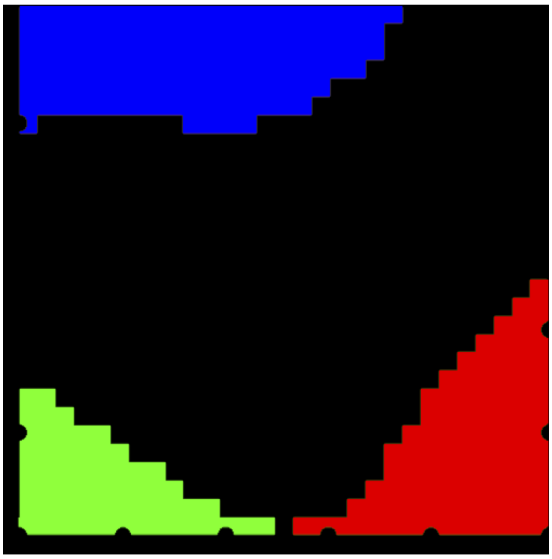

Figure 5.11. Image showing three blobs before horizontal merge

All three methods have their benefits and shortcomings. Whereas the bounding rectangle method is fast, it does not provide us with accurate location of the person. Calculating the centroid of the blob provides us with better approximation to the person's location, but it requires us to access all pixels in the blob and calculate the centroid, making it a slower method.

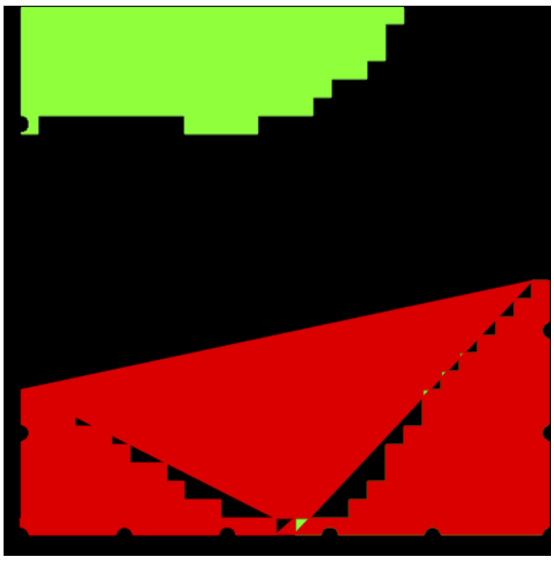

Figure 5.12. Horizontally merged blobs

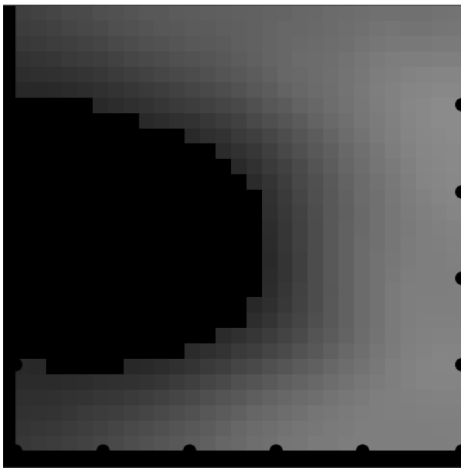

Figure 5.13. Image showing two blobs before merge

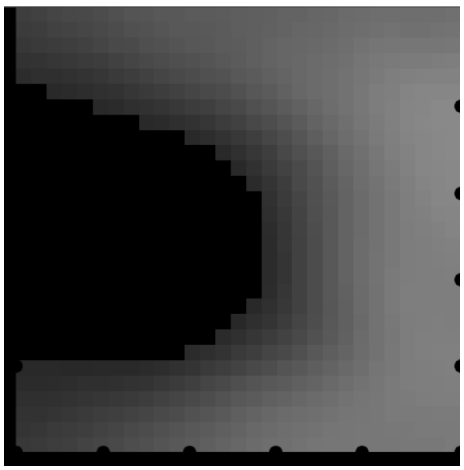

Figure 5.14. Image showing single merged blob

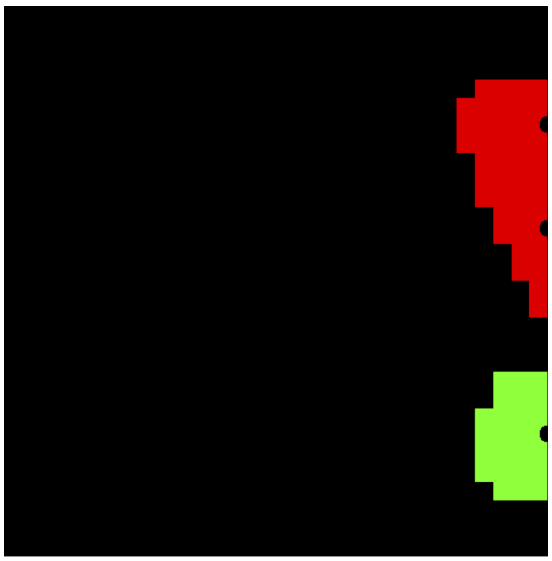

Figure 5.15. Image showing two blobs before split

Figure 5.16. Image showing blobs split apart

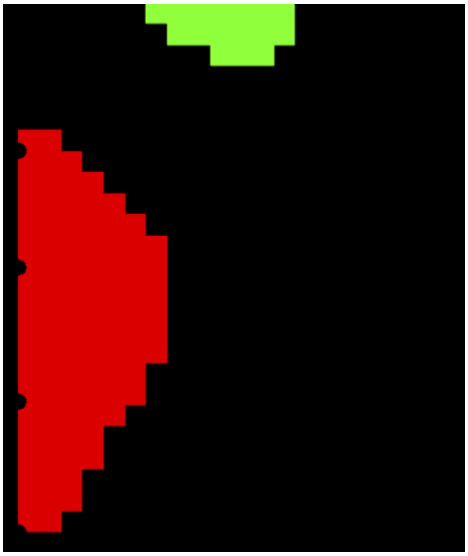

Figure 5.17. Image showing two blobs before noise filtering

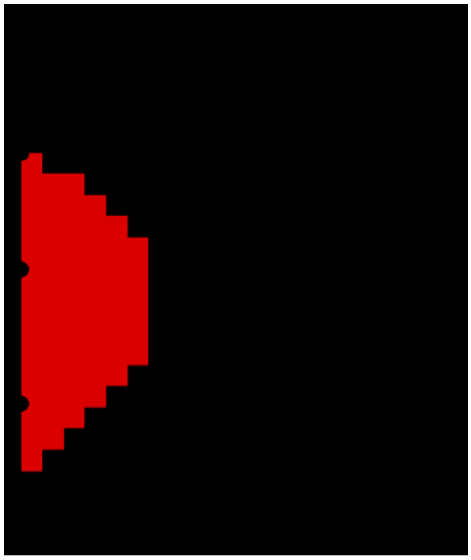

Figure 5.18. Image after noise filtering

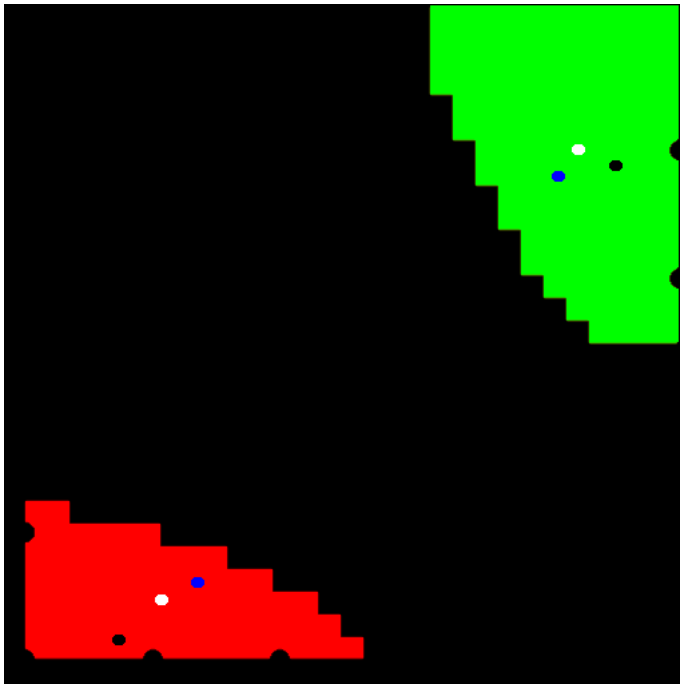

Figure 5.19. Image showing centers using three methods

CHAPTER 6

RESULTS

6.1 Results

In the following experiments, an image is considered to be misclassified if the number of blobs detected is not the same as the expected number of blobs. The expected number of blobs is known in the tests.

6.1.1 1-person Experiments

We found that the result is consistent across different experiments and not just a single 1-person experiment.

The total number of images used for the 1-person experiment in Table 6.1 is 3866.

The total number of images used for the 1-person experiment in Table 6.2 is 3852.

The total number of images used for the 1-person experiment in Table 6.3 is 3719.

The total number of images used for the 1-person experiment in Table 6.4 is 3714.

6.1.2 2-person Experiments

The total number of images used for the 2-person experiment in Table 6.5 is 1383.

Table 6.1. 1-person Images - Experiment 1

Method Name	Number of Misclassified Images	% of Misclassified Images
Fixed Threshold	139	3.59
Auto Adjust Threshold	1	0.02
Majority Count	1	0.02
Location based	0	0.0

Table 6.2. 1-person Images - Experiment 2

Method Name	Number of Misclassified Images	% of Misclassified Images
Fixed Threshold	527	13.68
Auto Adjust Threshold	42	1.09
Majority Count	0	0.0
Location based	0	0.0

Table 6.3. 1-person Images - Experiment 3

Method Name	Number of Misclassified Images	% of Misclassified Images
Fixed Threshold	282	7.58
Auto Adjust Threshold	26	0.69
Majority Count	25	0.67
Location based	0	0.0

Table 6.4. 1-person Images - Experiment 4

Method Name	Number of Misclassified Images	% of Misclassified Images
Fixed Threshold	214	5.76
Auto Adjust Threshold	5	0.13
Majority Count	1	0.02
Location based	0	0.0

Table 6.5. 2-person Images

Method Name	Number of Misclassified Images	% of Misclassified Images
Fixed Threshold	281	20.31
Auto Adjust Threshold	107	7.73
Majority Count	125	9.03
Location based	0	0.0

6.1.3 3-Person Experiments

The total number of images used for the 3-person experiment in Table 6.6 is 1011. Figure 6.1 shows graphical representation of the above tabular data.

For center calculation, based on our results, we see that centroid gives us the lease root mean square error (RMSE) across the experiments tested with it. This can be seen in Figure 6.2.

In order to further emphasize the importance of building history and using merge, split, update threshold, and noise operations, we counted the number of times we used them in our experiments. This can be seen in Table 6.7

We also wanted to check what impact saving the original image has compared to the

Table 6.6. 3-person Images

Method Name	Number of Misclassified Images	% of Misclassified Images
Fixed Threshold	858	84.86
Auto Adjust Threshold	799	79.03
Majority Count	799	79.03
Location based	445	44.01

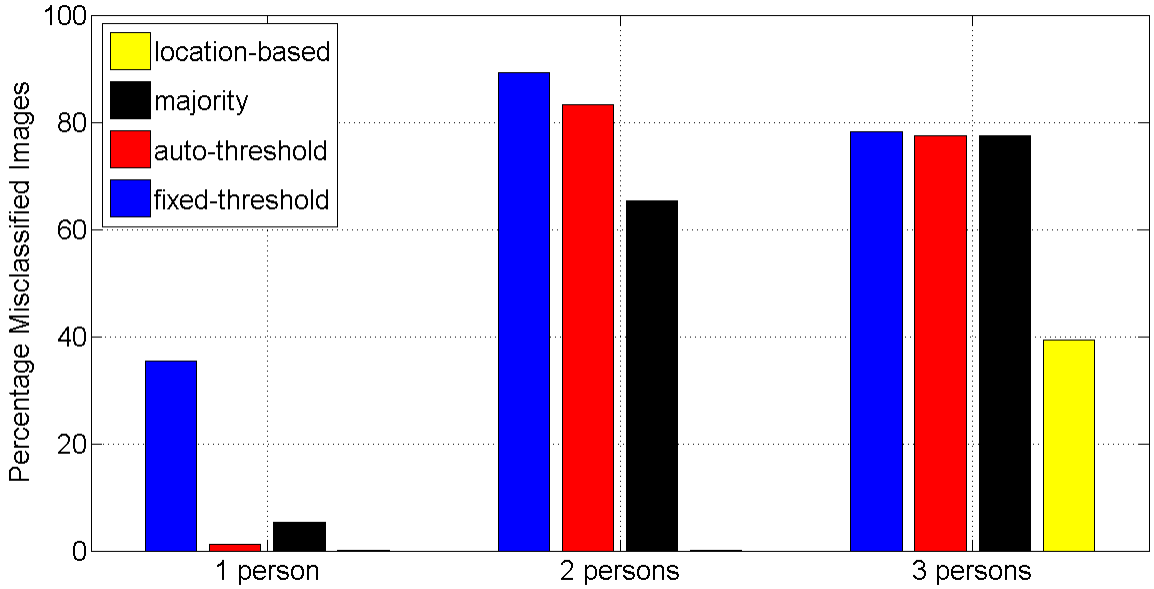

Figure 6.1. Image showing % misclassification rate for each method

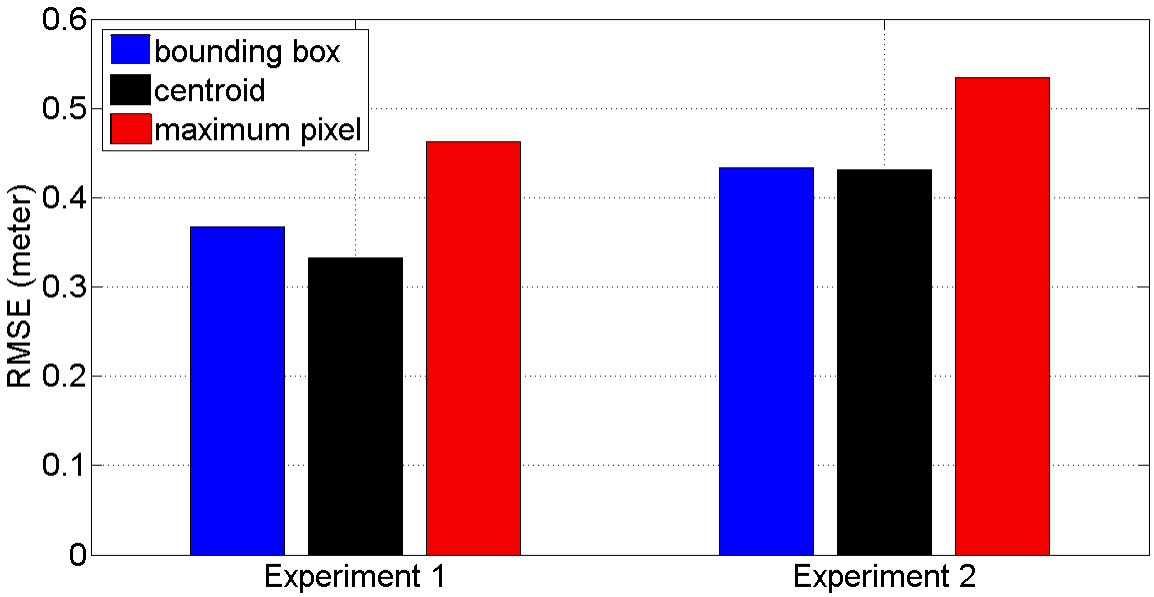

Figure 6.2. Image showing RMSE for center location methods

modified image on the detection accuracy. Table 6.8 shows the misclassification rate when we save the original image compared to when we save the finalized image after modification.

Table 6.7. No of Splits, Merges, Noise Declarations

Experiment	**NNo. of Merges**	**No. of Splits**	**No. of Noisy Blobs**
1-person	39	0	26
2-person	0	2211	0
3-person	0	109	0

Table 6.8. Modified vs Original % Misclassification Rate

Experiment	**Modified**	**Original**
1-person	0.0	20.55
2-person	0.0	72.51
3-person	39.34	65.16

CHAPTER 7

CONCLUSION

7.1 Conclusion

Our system uses Computer Vision techniques along with keeping track of historical information to accurately detect and track human motion in a monitored region. As part of this process, we identify the number of persons moving inside the region and calculate their location and track their respective motions. We also provide information about a new person entering the region or an existing person leaving the region. As part of the work, we have also shown why clustering-based approaches do not work well for this task. We also showed why spatial scan statistic does not work well for this task. We also discuss various approaches for detecting the number of persons in a monitored area, calculating their location, and then tracking them and show the differences, the benefits, and shortcomings of each of the methods. We also show the need of building a history of images which can be consulted with for various scenarios.

7.2 Future Work

- **More Results:** Run the method against more multiperson data and see how far we can push it. Right now, it was tested mainly against one person and two person data, which contain less noise. The real test would be against three and more person data.

- **RFID Tag Data:** The method should be extended to work with RFID tag data if available. This would improve the accuracy of the system. This would also make the system more resistant to noise as we can always rely on the RFID tag location and use that to give feedback to our system while running the tests.

- **Multistorey Setup:** Right now, the system can detect and track activities happening on the floor but for a multistorey building, this might not be sufficient. Therefore we need a system in which we can deploy some nodes on the roof and it might be able to do the same for the floor above. Also we should be able to deploy the nodes throughout the building and able to either get a 3-dimensional view of activities or

separate activity image for each floor without interference from the nodes on the floor below or the floor above.

- **Activity History:** The system should be extended to keep a history of past activities. This could be useful in elder homes. An elder might be performing the same tasks everyday. And if someday he does not do so or he is stationary at a certain location in the house, we would consider this deviation from the routine and can sound an alarm. This can be done without the history as well, but keeping this historical information could help make more accurate decisions.

- **Sophisticated Computer Vision:** We can use more sophisticated computer vision techniques to merge and split blobs. We use a simple approach for our tests by using a quadrilateral-based approach for merging and historical image information for splitting. This makes the task easier but might compromise on accuracy. Using better merging and splitting techniques for irregular sized objects could yield more accurate results in calculating Jaccard Similarity and hence increase the accuracy of the system.

REFERENCES

[1] F. Chang, C.-J. Chen, and C.-J. Lu, "A linear-time component-labeling algorithm using contour tracing technique," in *Computer Vision and Image Understanding*, vol. 93, 2004.

[2] C. Chang and A. Sahai, "Object tracking in a 2D UWB sensor network," in *38th Asilomar Conference on Signals, Systems and Computers*, vol. I, 2004.

[3] L. P. Song, C. Yu, and Q. H. Liu, "Through-wall imaging (TWI) by radar: 2-D tomographic results and analyses," in *IEEE Transactions on Geoscience and Remote Sensing*, vol. 43, no. 12, 2005.

[4] M. Youssef, M. Mah, and A. Agarwala, "Challenges: decide-free passive localization for wireless environments," in *Mobicom : ACM Int'l Conf. Mobile Computing and Networking*, 2007.

[5] J. Wilson and N. Patwari, "See-through walls: Motion tracking using variance-based radio tomography networks," in *IEEE Transactions on Mobile Computing*, vol. 10, no. 5, 2011.

[6] ——, "Radio tomographic imaging with wireless networks," in *IEEE Transactions on Mobile Computing*, 2009.

[7] T. Tsukiyama and Y. Shirai, "Detection of the movements of persons from a sparse sequence of tv images," in *Pattern Recognition*, vol. 18, 1985.

[8] D. C. Hogg, "Interpreting images of a known moving object," Ph.D. dissertation, University of Sussex, UK, 1984.

[9] R. Polana and R. Nelson, "Low level recognition of human motion," in *Workshop on Motion of Non-Rigid and Articulated Objects*, 1994.

[10] Y. Kameda and M. Minoh, "A human motion estimation method using 3-successive video frames," in *International Conference on Virtual Systems and Multimedia*, 1996.

[11] A. Nakazawa, H. Kato, and S. Inokuchi, "Human tracking using distributed video systems," in *International Conference on Pattern Recognition*, 1998.

[12] I. Haritaoglu, D. Harwood, and L. S. Davis, "W4 : Who? when? where? what? - a real time system for detecting and tracking people," in *International Conference on Automatic Face and Gesture Recognition*, 1998.

[13] T. Darrell, P. Maes, B. Blumberg, and A. P. Pentland, "A novel environment for situated vision and behavior," in *Workshop for Visual Behaviors at CVPR-94*, 1994.

[14] Y. Iwai, K. Ogaki, and M. Yachida, "Posture estimation using structure and motion models," in *International Conference on Computer Vision*, 1999.

[15] S. Iwasawa, K. Ebihara, J. Ohya, and S. Morishima, "Real-time estimation of human body posture from monocular thermal images," in *Conference on Computer Vision and Pattern Recognition*, 1997.

[16] R. Eshel and Y. Moses, "Homography based multiple camera detection and tracking of people in a dense crowd," in *Computer Vision and Pattern Recognition*, 2008.

[17] M. Yamada, K. Ebihara, and J. Ohya, "A new robust real-time method for extracting human silhouettes from color images," in *International Conference on Automatic Face and Gesture Recognition*, 1998.

[18] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland, "Pfinder:real-time tracking of human body," in *Trans. Pattern Analysis and Machine Intelligence*, vol. 19, 1997.

[19] J. B. Kim and H. J. Kim, "Efficient region-based motion segmentation for a video monitoring system," in *Pattern Recognition Letters*, vol. 24, 2003.

[20] C. C. Chang and C. J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.